AFRL-IF-RS-TR-2004-279
Final Technical Report
October 2004

ATTEND: ANALYTICAL TOOLS TO EVALUATE NEGOTIATION DIFFICULTY

Information Sciences Institute, USC

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K277

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-279 has been reviewed and is approved for publication

APPROVED: /s/
DANIEL E. DASKIEWICH
Project Engineer

FOR THE DIRECTOR: /s/
JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 2004	**FINAL** **May 02 – Dec 03**

4. TITLE AND SUBTITLE

ATTEND: ANALYTICAL TOOLS TO EVALUATE NEGOTIATION DIFFICULTY

5. FUNDING NUMBERS

G - F30602-00-2-0533
PE - 62301E
PR - ANTS
TA - 00
WU - 04

6. AUTHOR(S)

Alejandro Bugacov
Robert Neches

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Information Sciences Institute, USC
4676 Admiralty Way, Suite 1001
Marina del Rey CA 90292-6695

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFTB
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2004-279

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Daniel E. Daskiewich/IFTB/(315) 330-7731 Daniel.Daskiewich@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

Purpose: The ATTEND (Analytical Tools To Evaluate Negotiation Difficulty) project was established to study computational complexity issues arising in complex, dynamic and large-scale real-world problems requiring finding "good-enough/soon-enough" assignments of resources to tasks. "Good-enough/soon-enough" problems arise in situations where finding the best solution obtainable within time limits was preferable to finding an optimal solution in unbounded time. Examples range from real-time fire control problems in which time is of the essence, to operational risk management and logistics problems in which size and complexity make it computationally infeasible to seek full optimality.

Scope: The effort focused upon flight operations scheduling problems exemplifying challenges faced by Marines Corps flight schedulers for AV8-B-Harrier aircraft in Marine Aircraft Group 13.

Methods: Our approach mapped resource allocation, planning and scheduling problems to formal declarative representations which could then be solved and characterized using available state-of-the-art constraint solvers.

Major Findings, Including Results, Conclusions, and Recommendations: ATTEND showed the effectiveness of a multi-phase hybrid approach to solve computationally hard real-world problems: combining multiple solvers proved to be orders of magnitude more efficient that any individual solver that has been built or proposed for problems of the class which was studied.

14. SUBJECT TERMS

Resource Allocation, Negotiation Technology, Planning and Scheduling, Propositional Reasoning, Computational Complexity

15. NUMBER OF PAGES 23

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

1 Summary

The main focus of the ATTEND (Analytical Tools To Evaluate Negotiation Difficulty) project was the study of computational complexity issues arising in complex and large-scale real-world logistics problems and the implementation of efficient solvers in the context of its companion ANTS (Autonomous Negotiating Teams) program project, CAMERA (Coordination and Management Environments for Responsive Agents), and the current application of CAMERA to safety-oriented flight scheduling, Schedules Negotiated by Agent-based Planners (SNAP).
Our approach is based on mapping the underlying combinatorial optimization problems (resource allocation, planning and scheduling) to formal declarative representations that can then be solved and characterized using available state-of-the-art constraint solvers.
We concentrated our efforts in the analysis of the flight scheduling problems arising in SNAP, the current application of the ANT's CAMERA project. The solution of these problems represent a real challenge in the fields of AI and operations research due to the fact that they combine contingency planning and scheduling for long planning horizons (weeks to several months) at very fine (1 minute) time resolution precision.
During the life of the project we mainly focused on the encoding of these hard problems into Boolean Satisfiability (SAT) and in Pseudo-Boolean (PB) overconstrained integer programming representations that enable the use of general-purpose state-of-the-art engines to solve the problem and characterize their complexity based solely on the properties of the encoded problem.
We developed three main types of solvers for the SNAP planning and scheduling problem: SAT, Pseudo-Boolean and a Hybrid Two-Phase Solver. Each of these solvers were tied to different representations of the problem with increasing fidelity in terms of modeling the large and complex set of constraints of the SNAP domain. These solvers were implemented and integrated to the basic negotiation architecture developed by the CAMERA project. This integration showed the effectiveness of the multi-phase hybrid approach to solve computationally hard real-world problems in the logistics domain. This basic architecture enables the implementation of procedural and declarative methods that combined proved to be orders of magnitude more efficient that any individual solver that has been built or proposed for the domain. This technology was demonstrated on June 2003 at DARPA (Defense Advanced Research Projects Agency) in an exercise combining the efforts of the logistics and other complexity and dynamics contractors in the ANTs program. In this demonstration we showed the creation of a hard-to-solve tightly-packed three day surge schedule that required some sophisticated scheduling and planning techniques in order to schedule all missions. We showed how the composition of a two-phase hybrid solver was able to find the correct schedule in less than 4 minutes while any of the individual solvers would fail to find a solution regardless of the time or require orders of magnitude more of execution time to solve.

2 Introduction

The main focus of our research was on the study and characterization of the computationally hard constraint optimization problems arising in SNAP, the current application of the companion ANTs project CAMERA to safety-oriented flight scheduling SNAP. These problems combine contingency planning and scheduling for long planning horizons (weeks to several months) at very fine (1 minute) time resolution precision and their solution represent a real challenge to existing constraint programming and AI search techniques.

During the project we introduced and explored four main ideas to solve and analyze these computationally-hard problems:

- Adaptive control of negotiation and problem reformulation enabled by analytical methods to estimate their negotiation difficulty
- Management of resource contention facilitated by analysis of constraint-based encodings of scheduling and planning problems
- Complexity reduction via phase-transition aware reformulation of the problem
- Development of hybrid multi-phase solvers in order to solve problems that cannot be solved in reasonable time by any individual solver

The main technical challenges of the problems in the logistics domain that we studied can be summarized as follows:

- Large combinatorial constraint optimization problem that involves planning & scheduling at wide range of time-scales
- Scheduling of flight missions with multiple complementary requirements
- Implicit task dependency via enablement of resources skill level and goals
- Several types of resources: consumable, re-usables, dynamic skill level, relocatable
- Mix of continuous (e.g., time) and finite-domain (resources) variables
- Complex hard-to-express constraints and objectives:
    - Crew day Restrictions
        - Global constraints that cut across tasks
    - Squadron Capability
        - Mix of qualifications that should be represented in the squadron
    - Average Pilot CRP (combat readiness percentage)
        - CRP is a fractional value for mission codes
    - Equity in Pilot Hours
        - E.g., within ~10% equity in a monthly basis

In order to model and solve these problems we adopted declarative representations of the problems and integrated them into CAMERA's basic architecture. Among the main advantages of these declarative methods are the availability of state-of-the-art search engines being constantly improved by a very active research community, the possibility of using soft and hard

constraints which are well suited for balancing multiple objectives and doing optimization, and the capability of defining different search schemes like local search or complete search to find the best possible solution, optimal solutions or bounds to the problem. Furthermore, the size of the resulting encodings can be used to characterize the complexity of the problems.

By integrating these advanced constraint programming tools into CAMERA's architecture, our goal has been to guide the reformulation of SNAP's real-world scheduling problems based on problem complexity analysis and good-enough/soon-enough constraints such that we can reformulate problems into a solvable (but still interesting) subset, indicate when to compromise quality in order to meet time constraints and be able to identify difficult-to-find solutions.

3 Methods, Assumptions and Procedures

There are two major trends in methodologies to solve complex combinatorial optimization problems. One is building specialized solvers containing sophisticated heuristics derived from precise knowledge of the domain. The other method is to encode the problem into a set of variables and constraints and use general state-of-the-art search techniques to find the assignments of variables that satisfy or optimize the system of constraints. This second approach can be competitive and sometimes outperform the specialized solvers. It also provides some additional long term benefits in terms of maintenance of the solver and performance. However, the burden of the overall efficiency of the solver may sometimes be on finding a proper encoding of the problem.

The last ten years have seen tremendous advances in the efficiency of SAT solvers and they can now solve systems with up to 10^5 variables within minutes. This has motivated some researchers to extend these advanced search techniques to other similar representations that are more appropriate for certain types of problems. Of particular interest to our problem is the Pseudo-Boolean (PB) representation. The encoding of this problem results in a very large number of cardinality constraints (e.g., the ones associated to exclusion of a resource to be assigned to more than a mission at a time) that can be written in PB in a form that is much more compact than its equivalent in SAT. In addition to the compactness of the encoding in the PB formalism, these solver engines also provide for optimization capabilities. This makes these solvers very attractive for our domain of real-time resource allocation where in most situations the users are interested in finding the best possible schedule that can be computed within a given execution time. In the PB formalism, constraints are expressed as linear inequalities over a set of integers variables with domain (0,1). Constraints are weighted and can be either hard or soft. The PB engine searches for an assignment of variables that satisfies all hard constraints and minimizes the sum of the weights of violated soft constraints. Since PB is a much more expressive and compact representation than SAT for problems involving arithmetic reasoning and there are very efficient general-purpose PB solvers available, we adopted this technique to solve a time-discretized model of the SNAP scheduling and planning problem.

During the ATTEND project we developed three main types of solvers for the SNAP planning and scheduling problem: SAT, Pseudo-Boolean and a Hybrid Two-Phase Solver. Each of these

solvers were tied to different representations of the problem with increasing fidelity in terms of modeling the large and complex set of constraints of the SNAP domain.

In the SAT solver we considered a model problem that only captured the resource allocation part of the scheduling problem, ignoring time and considering the assignment of resources to a set of tasks with multiple complimentary requirements. We refer to this problem as the *Marbles* problem. This formalism was a useful initial step towards understanding the complexity of that simple model and to derive complexity profiles that enabled the quick prediction of the maximum number of tasks based solely in the characteristics of the encoded problem. However, because it was based on a satisfiability model it was hard to do optimization and the size of the encodings was excessively large to treat large problems due to the cardinality constraints.

Next we moved to the solution of simplified model of the SNAP scheduling problem, i.e., we introduced time to the previous Marbles model. In order to use declarative methods, we discretized time by dividing the planning horizon into a set of equally-spaced time slots and implemented a sophisticated methodology where the problems can be discretized at variable time resolutions (from 1 minute to days). In order to be able to do constraint optimization and to avoid the large number of SAT clauses deriving from the cardinality constraints, we adopted the Pseudo-Boolean representation to encode this new model of the problem. In the PB representation we can associate soft-constraints to the scheduling of a task or other quantities and optimize the number of scheduled tasks or the sum of the tasks priorities. We used a state-of-the-art general purpose PB engine called OPARIS (developed by CIRL contractor of the ANTs program) to find the variables assignment to the resulting encoding of the problem.

Although the PB solver is highly-sophisticated in terms of backtracking capabilities to obtain hard-to-find solutions and reasonably fast (it can solve a 3-day schedule in a few minutes at 30 minutes resolution), this model of the problem doesn't consider all constraints and would take too long to solve problems at the required 1 minute resolution. To solve that problem we implemented a two-phase hybrid solver that combines the search power of the PB solver to find a partial solution to the *difficult* part of the problem in its first phase with the speed of a SNAP greedy solver that validates additional constraints and schedules at a finer time resolution in the second phase. The main steps involved in this two-phase solver are shown in Figure 1.

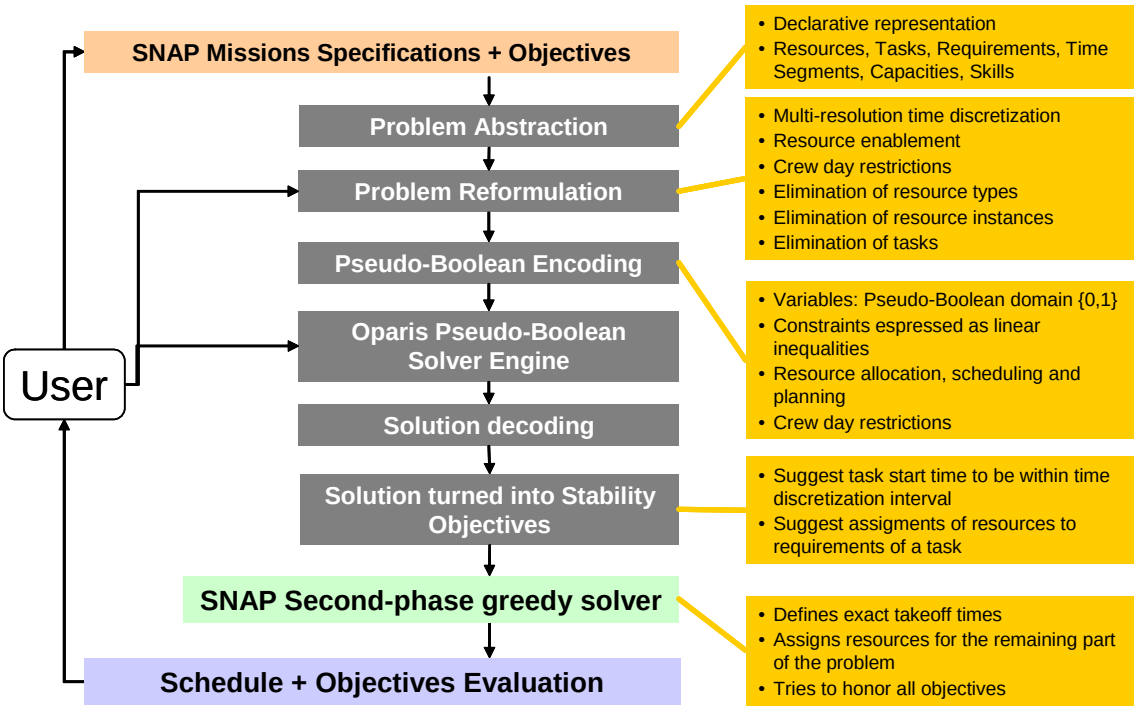

Figure 1: Hybrid two-phase Solver: Problem Abstraction and Reformulation

The following sub-section describes in great detail the PB encodings of the time-discretized version of the SNAP scheduling problem.

a. Declarative representation of the SNAP Scheduling and Planning problem

i. Problem Definition

In order to solve the SNAP Scheduling problem using sophisticated general-purpose search engines we introduced a time-discretized model of the problem that captures the most relevant features of the domain but is general enough that results obtained using this model problem can be useful in other domains sharing the main features of the scheduling problem. To use finite domain encoding techniques we discretize time by dividing the length of the planning horizon, L; in a finite set of S time slots of equal duration w. In our implementation the problem can be discretized at variable time-resolution with time-step w >=1 minute. The same time-step is used to discretize all time intervals of the tasks and availability of resources. In the problem definition all time intervals are described as an Availability Slot Range which is represented as:

- Start slot: The first slot in the range
- Length: Number of slots in the range
- Capacity: Capacity during that time interval

Thus, a problem is characterized by the following entities:

- T = {T_1, T_2,….,T_N}: The set of N Tasks
- R = {R_1, R_2,….,R_M: The set of M Resources
- L : The duration of the planning horizon
- w : The duration of the time-discretization step
- s0 : The time (date) of the beginning of the planning horizon

The dependencies between all entities of a problem can be seen in Figure 2.

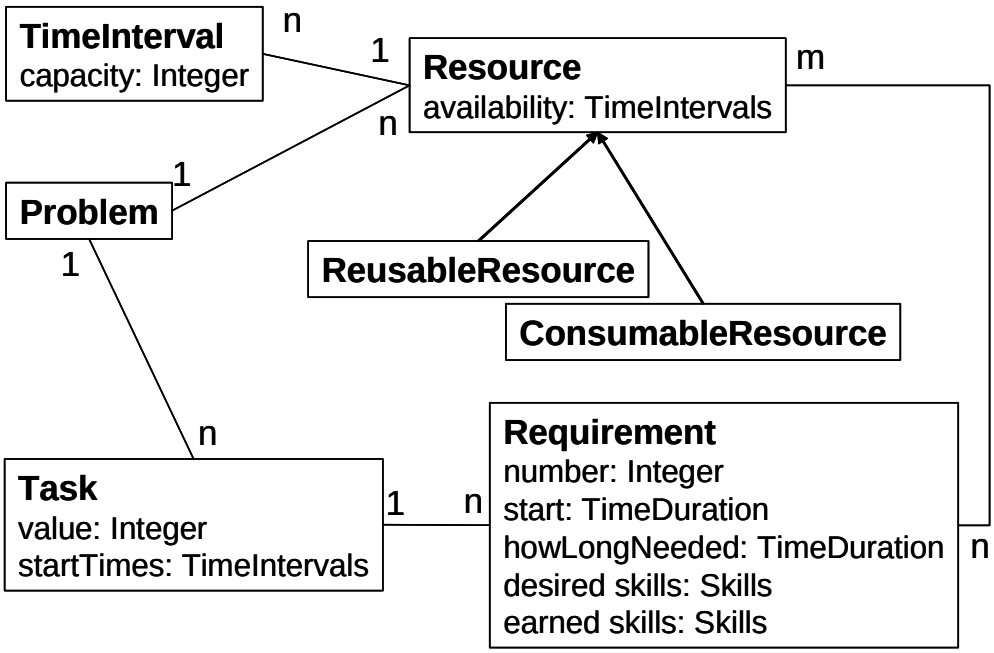

Figure 2: Problem abstraction declarative representation

In Figure 3 we sketch an example of the definition of a task with 6 requirements in a problem with 10 resources and a planning horizon of 21 slots:

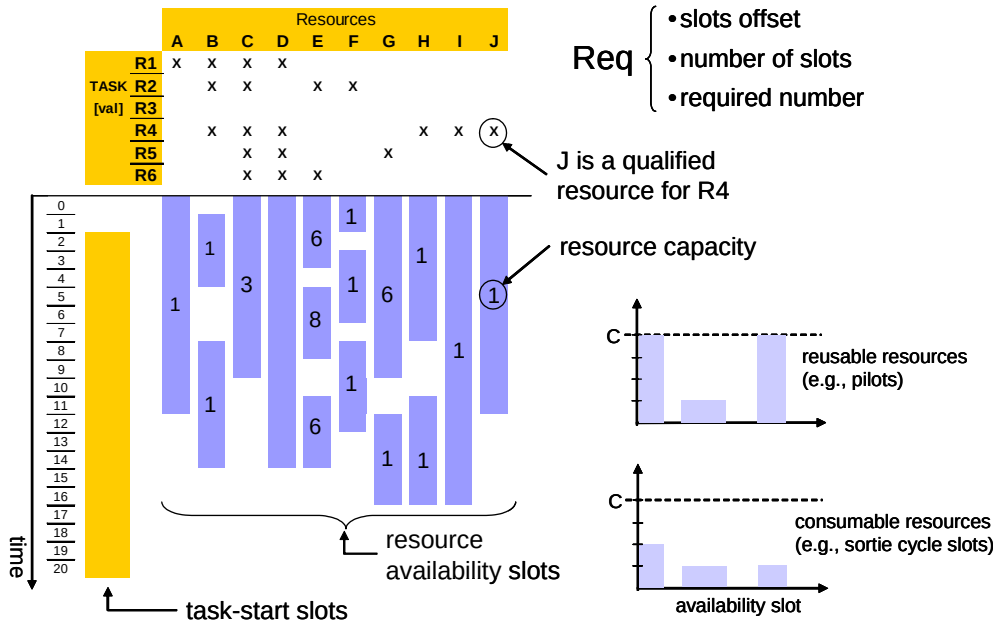

Figure 3: Time-discretized SNAP problem

Each Resource in the set R is described by:

- Name
- Resource Type (E.g., Pilot, Aircraft, Simulator)
- A set of Availability Slot Ranges. This set represents the times during which a resource is available and its capacity.

ii. Time-discretization of tasks and resources to enable multi-resolution scheduling

The time discretization divides the planning horizon into a set of time slots of equal duration. We refer to this duration as the resolution of the encoding. The resolution represents the smallest measure of time that can be considered in the resulting schedule. For example, if the time resolution is 1 hour, we can only determine the start time of a task with 1 hour precision. The goal of the introduced time discretization scheme is to enable the modeling and encoding of the problem at variable time resolution, from a few minutes to 1 day. This goal introduces several complications to the model. For example, let's imagine that we discretize at 1 day resolution and that a pilot is available for a period of 8 hours during a given day, and that would enable him to fly in two missions of 4 hours each. Now, when we discretize to 1 day, both the duration of the tasks and the availability of the pilot is reduced to a single slot. This generates the issue of how to use a pilot for more than 1 mission using this 1 day resolution.

To address this problem we introduced fractional capacities and fractional required amounts of a resource for each requirement. This allows us to scale down the required amount of a resource that a task requires and therefore, in the previous example, the task needed 1 amount of a pilot for each 4 hours mission, now the capacity of the pilot resource will still be of 1 but the task will require only 0.5 of a pilot and therefore a pilot would be able to fly in two missions during a given day. Figure 4 depicts how the discretizing scheme works for long (1 day) and short (2 hours) durations of the time discretization slot.

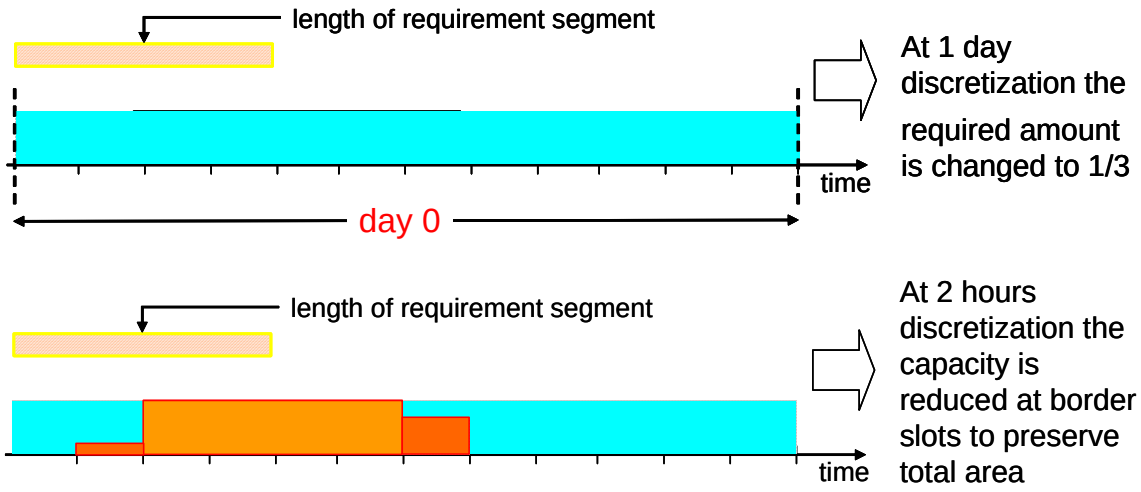

Figure 4: Discretization of tasks and resources segments

iii. Pseudo-Boolean Encoding

In the Pseudo-Boolean (PB) formalism, constraints are expressed as linear inequalities over a set of integer variables with domain {0,1}. Constraints are weighted and can be either hard or soft. The PB engine searches for an assignment of variables that satisfies all hard constraints and minimizes the sum of the weights of violated soft constraints.

In the following subsections we describe the variables and types of constraints that we needed to introduce in order to obtain high quality, valid solutions of the model problem that can then be used to generate full solutions of the SNAP scheduling problem.

4 Variables

In our encoding we introduced a hierarchy of variables where top-level variables act like switches or activation variables for the low-level variables. These low level variables are the ones that actually book or schedule the resources for a given task-requirement pair at a given time slot of the planning horizon. The activation variables or switches have the characteristic that when they are turned off (i.e., evaluated to 0) the constraints in which they appear are trivially satisfied.

For the basic set of constraints we introduce the following 4 types of variables:

- T_i: Task Activation Variables
- T_ik :Task Starting-Slot Activation Variables
- X_i,j: Resource Task Requirement Variable. Represents the assignment of Resource X to Requirement j of Task i. Are used both for Reusable and Consumable Resources.
- X_i,j,k: Reusable Resource Slot Variable. Represents the assignment of Slot k of the Reusable Resource X to the requirement j of task i.

These variables are organized in a hierarchical structure as shown in Figure 5.

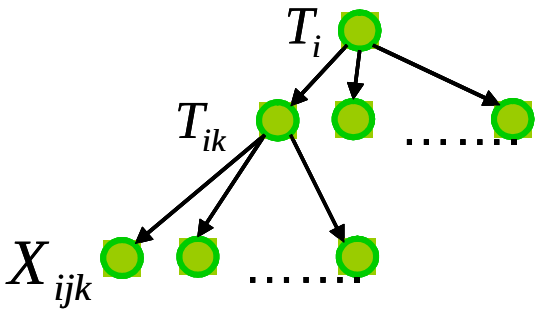

Figure 5: Pseudo-Boolean variables hierarchical structure

Soft Constraints

Currently we only introduce soft constraints to activate the top-level task activation variables. We introduce one soft constraint per task and weight it with the task value so that the solver will try to find the solution that maximizes the sum of the value of the scheduled tasks.

$$[soft] \quad V_i \, T_i \; \geq 1 \qquad for \;\; i = 1, 2, \ldots, N.$$

In our implementation we have some options that flatten the task values so that the solution will correspond to the scheduling of the maximum number of tasks.

Task-Start Slot Selection Constraints

We introduce one of these constraints per task. They are used to select at least one of the possible task starting slots if the task's activation variable is turned on and none otherwise.

$$\overline{T}_i + \sum_{k \in \mathbf{S}_{T_i}} T_{ik} = 1 \quad for \;\; i = 1, 2, \ldots, N.$$

Here S_{T_i} indicates the set of slots k that are feasible starting slots for the task considering the offsets and length of the segments for all requirements in the task i.

Task-Requirement Resource Selection Constraints

Once a starting slot has been selected by the constraints above, we use the following constraints to pick a resource among the set of possible resources for that starting slot k. At this point we don't distinguish between what instances of resource X we are selecting. We only worry that in its initial availability intervals the resource has enough capacity to accommodate that requirement.

$$\overline{T}_i + \sum_{k \in \mathbf{S}_{T_i}} T_{ik} = 1 \quad for \;\; i = 1, 2, \ldots, N.$$

In this equation we should include the values of k that are feasible starting times for that task. In the previous expression, it is possible to select more than one resource for that requirement of a task. To exclude that possibility we introduce the following additional condition

$$\overline{T}_i + \sum_{X \in \Omega_{ij}} X_{i,j} = 1 \;\; for \;\; \begin{array}{rcl} i & = & 1, 2, \ldots, N \\ j & = & 1, 2, \ldots, r_i \end{array}$$

Reusable Resources Slots Selection Constraints

These constraints are used for reusable resources only to turn on d_ij consecutive slots (of its availability interval).

$$d_{ij}\,\overline{T}_{ik} + d_{ij}\,\overline{X}_{i,j} + \sum_{l=0}^{d_{ij}-1} X_{i,j,k+s_{ij}+l} \;\geq\; d_{ij}$$
$$i \;=\; 1, 2, \ldots, N$$
$$j \;=\; 1, 2, \ldots, r_i$$

This condition alone will, in principle, enable some spurious slots from the resource availability to be also turned on. To avoid this effect we impose an additional condition to say that we don't want more than d_ij slots on for a selected resource and we want all slots turned off if that resource wasn't selected.

$$d_{ij}\,\overline{X}_{ij} + \sum_{k \in \mathbf{K}(i,j)} X_{i,j,k} \;\leq\; d_{ij}$$
$$i \;=\; 1, 2, \ldots, N$$
$$j \;=\; 1, 2, \ldots, r_i$$

Where K(i,j) is the whole set of feasible slots for the scheduling of resource X to the requirement j of task i, taking into consideration the initial resource availability and capacity of X, the possible task start slots for task i and the segment length, offset and required capacity of requirement j.

Capacity Exclusion Constraints

In order to have correct solutions we need to preclude the assignment of a resource slot to more requirements than its initial capacity. Since reusable and consumable resources are modeled differently in terms of their slots assignments (i.e., we don't represent consumable resources at the atomic slot level, but only at the range availability level) we have two different expressions for reusable and consumable resources.

$$\sum_{i,j \in X(i,j,k)} n_{ij} X_{i,j,k} \;\leq\; c(X,k) \;\; for X \;\in\; \mathbf{R}_{reusable}$$
$$\sum_{i,j \in X(i,j,k)} n_{ij} X_{i,j,k} \;\leq\; c(X) \;\; for X \;\in\; \mathbf{R}_{consumable}$$

Where X (i,j,k) is the set of task-requirement pairs in which X is a possible resource for start-slot k.

Crew Day Constraints

For a given problem we can specify a particular kind of crew day constraints via the following parameters:

- c: crew day length (e.g., 12 hours)
- p: period (e.g., 24 hours)
- s: maximum shift per day (e.g., 2 hours)
- s_total: maximum total shift (e.g., 20 hours)
- γ: maximum number of tasks per crew day
- m: number of ``crew day slots" per slot (positive integer)

In addition, the problem specifies what kind of resources must adhere to crew day rules (typically pilots). The rest of this section describes encoding the crew day constraints for a single resource, a pilot. The same encoding is replicated if there are multiple pilots.
All times and durations in this section are measured in ``crew day slots," which are integer multiples of the slots used for resource availability. If the problem is discretized to 10 minutes for resource availability, crew day slots would be \$10\cdmultiple\$ minutes.

The crew day constraints can be summarized as follows:

- $-c < -c_0 - \leq p - c$
- $(\forall l)\ p - s \leq c_{l+1} - c_l \leq p + s$
- $(\forall l)\ - \leq c_l - c_0 - lp \leq$
- No work happens outside of a crew day, and no pilot does more than γ tasks in a single crew day.

In order to enforce these constraints, we introduce the following pseudo-Boolean variables:

$d_{l,b}$	one bit of the difference $c_l -$
A_k	true iff pilot is at rest in slot k
	true iff crew day l contains slot k
$Y_{i,k}$	true iff pilot does task i in slot k
	true if (not iff) this pilot does task i on crew day l

The quantity c_l is not a pseudo-Boolean variable, but it may be expressed as:

$$-2^n d_{l,n} + d_{l,0} + 2d_{l,1} + \cdots + 2^{n-1} d_{l,n-1}$$

where the value of n is determined by how large a range must be representable.

Given that, we can trivially encode

$$\begin{aligned}
&-c < -c_0 - \leq p - c, \\
&p - s \leq c_{l+1} - c_l \leq p + s, \\
&- \leq c_l - c_0 - lp \leq, \text{ and} \\
&(c_l >) \rightarrow (c_l - c_{l-1} = p).
\end{aligned}$$

The last of those is merely to reduce the number of redundant, equivalent solutions.

5 Collaborations

During the project we collaborated with other teams of the ANTs community.

With Bart Selman and his co-workers from Cornell University, we collaborated in developing a modular SAT encoding that resulted in a phase-transition-based prediction mechanism for the Marbles 1 problem and in implementing a planning extension to ATTEND's declarative encoding of the scheduling problem.

With Andrew Parkes and his co-workers from the Computational Intelligence Research Laboratory (CIRL) of the University of Oregon we worked on the integration of the OPARIS pseudo-Boolean (PB) engine to ATTEND's PB solver.

We also collaborated with the team from Altarum in their experimentation to improve dynamics robustness of the SNAP system. And we collaborated with Weinxion Zhang from Washington University of Saint Louis on their complexity experimentation of Marbles problems and we provided them with benchmark problems for their backbone guided search engine.

6 Results and Discussion

During our work in the logistics domain of the ANTs program, we have developed solvers and integrated them to CAMERA's basic architecture to demonstrate the efficiency of the hybridization approach to solve computationally hard real-world problems in the logistics domain. This basic architecture enables the implementation of procedural and declarative methods that combined proved to be orders of magnitude more efficient that any individual solver that has been built or proposed for the domain. Although the composition of this efficient

hybrid solver was mostly done through a laborious and time-consuming trial and error process, we have sufficient evidence to believe that the process can be automated and optimized for this class of problems in the domain.

This technology was demonstrated in June 2003 at DARPA in an exercise combining the efforts of the logistics and some complexity and dynamics contractors in the ANTs program. In the demo we showed the creation of a hard-to-solve tightly-packed three day surge schedule that required some sophisticated scheduling and planning techniques in order to schedule all missions. We showed how the composition of a two-phase hybrid solver was able to find the correct schedule in less than 4 minutes. Any of the individual solvers would fail to find a solution regardless of the time or require orders of magnitude more of execution time to solve it.

In the first phase of this hybrid solver, we simplified the problem by discretizing it to ½ hour resolution and removed the range resources from consideration. The Pseudo-Boolean solver was able to compute a solution in 3 minutes using a 2.4 GHz processor. The encoding of the problem consisted of about 70K variables and 200K constraints.

In the second phase of the hybridization process, we encoded the solution to the simplified problem as soft-constraints for the second-phase solver. The solver selected for the second-phase was a greedy solver that has complete knowledge of the domain and can very rapidly test the validity of an assignment but lacks the sophisticated backtracking capabilities necessary to obtain hard-to-find solutions to the overall problem. With the soft-constraints provided by the PB solver, the greedy solver was able to fill the rest of the requirements for each task and produce a 1-minute resolution solution within 15 seconds.

The demonstration showed the feasibility of the hybrid approach for solving seemingly intractable problems. We believe this constitutes the main result and contribution of the project.

Other results and lessons learnt during the project can be broken into the categories discussed in the paragraphs following.

Nature of the problem and encoding issues.

It is hard to combine continuous (time) with discrete variables (tasks + resources). It might be worthwhile to investigate in mixed solvers. It is also hard to implement a robust time-discretization scheme. We need to introduce cutoff regions to modify the way the encoder treats the time slots. As shown in Figure 7, the time discretization scheme has sweet-spots that make solution-quality sensitive to some special values of the time slot duration. The time discretization introduces resource and requirements capacities that greatly complicate the encodings. Further research into finite domain methods might be needed in order to study trade-off between more compact representation and search engines.
Crew day introduces global constraints that cut through all tasks, though they are hard to encode and, as shown in Figure 7, they increase the total number of constraints by an order of magnitude.

The more difficult part of the encoding is that it combines planning and scheduling. (The resource enablement issue necessitates introducing planning to the problem.) It is hard to create solvers that are better than left-to-right greedy solvers. Level of detail is hard to work with 1 minute precision in long planning horizons of weeks and months. We need to introduce hybrid solvers in order to treat the problems.

The Pseudo-Boolean framework is very useful for the domain.

Due to cardinality constraints pseudo-Boolean is much more suitable than SAT to encode these types of problems. The resulting encoding is much more compact, and it is easier to do optimization. We can define soft and hard constraints and we can still count variables and constraints--although this does not necessarily translate into counting variables and clauses. Nevertheless, it can still be used as an indicator of the complexity of the problem.
We had good results from using switches or activation variables in the encoding. Variables hierarchy can be used by solvers (e.g., OPARIS from CIRL) to efficiently prune the search space.
Soft-constraints were very useful but complicated for some solvers. WSATOIP is very sensitive to the ratio of soft to hard constraints.
Solvers can be very sensitive to the parameters set. We need to build experience to tune them appropriately

Open issues in Complexity Research.

The difficulty of one of our problems can be greatly increased by changing/adding one constraint (e.g., a hard constraint on the number of tasks scheduled).
Work on random problems may be difficult to apply to the problems we care about. However, by measuring lots of problems and their PB encodings, we can get some idea of the expected running time.
It remains a topic for further research to fully understand how information learnt about an ensemble of problems can be dynamically applied to solving specific problems.

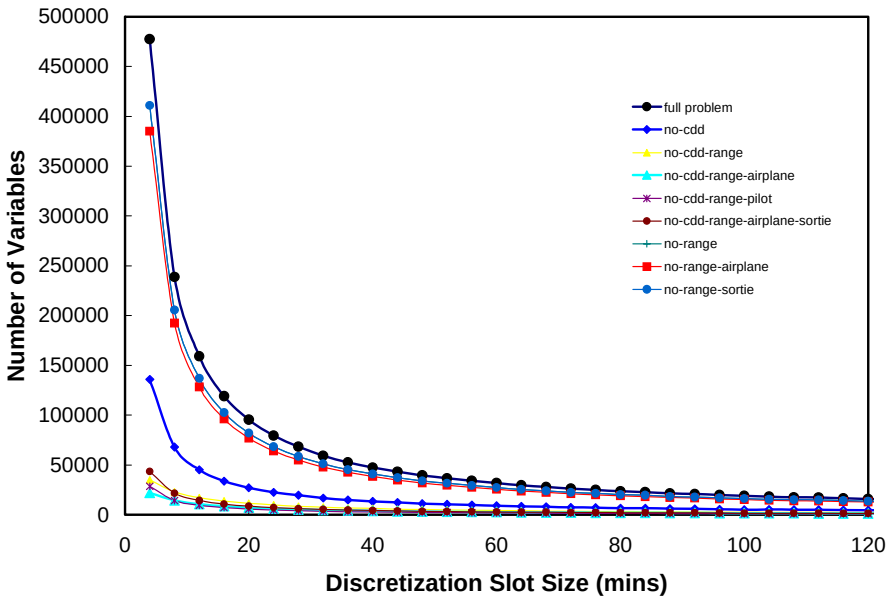

Figure 6: Crew day constraints dominates number of variables

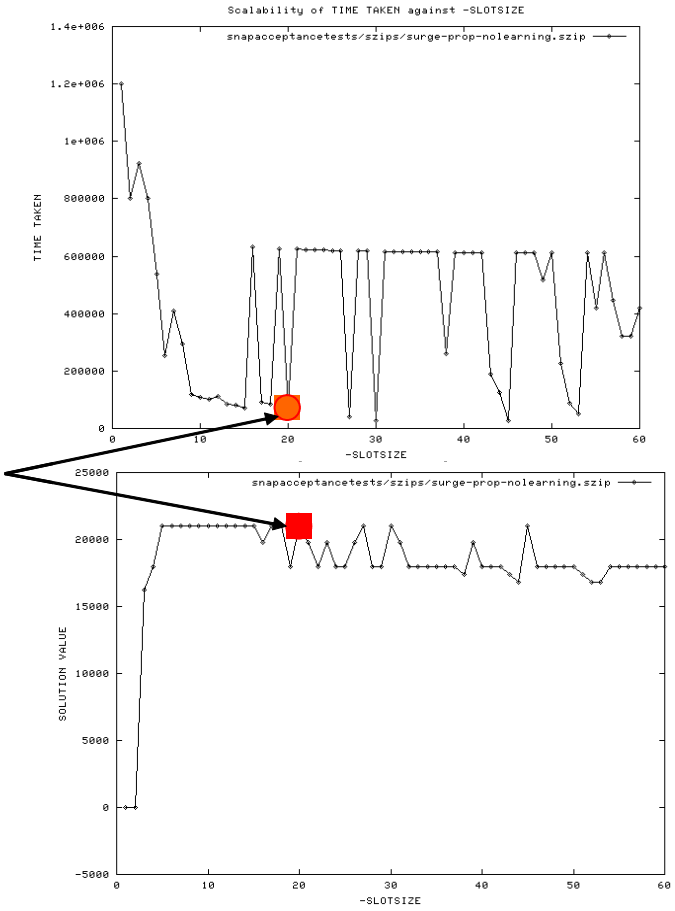

Figure 7: Time-disctretization sweet spots

7 ANTs eBook editing/hosting

During the last phase of the project, we played an active role in the web hosting and editing of the ANTs ebook. The eBook is an online collection of the most relevant work produced by the contractors of the ANTs program. It is currently available online at http://www.isi.edu/~szekely/antsebook/ebook/ .

Bibliography

Distributed Resource Allocation: Knowing When To Quit. Jinbo Chen, Alejandro Bugacov, Pedro Szekely, Martin Frank, Min Cai, Donghan Kim and Robert Neches. Accepted at the Representations and Approaches for Time-Critical Decentralized Resource/Role/Task Allocation Workshop of the Second International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2003). Melbourne, Australia

A. Bugacov, P. Szekely, G. Pike, D. Kim, C. Domshlak, C. Gomes and B. Selman. Multi-Resolution Modeling and Pseudo-Boolean Encoding of the SNAP Scheduling Problem. Included in the ANTs eBook.

Alejandro Bugacov, Aram Galstyan and Kristina Lerman. Threshold behavior in a Boolean network model of SAT. Paper submitted to the 2003 Int'l MultiConference in CS + CE + Math (IC-AI'03) Las Vegas, June 23, 2003.

Alejandro Bugacov, Donghan Kim, Carla Gomes and Bart Selman. SAT Encoding of a Resource Allocation Problem with Modular Constraints. Paper submitted to the Sixth International Conference on AI Planning & Scheduling (AIPS02), April 2002.

Bugacov, A., Kim, D., Gomes, C., and Selman, B. "Modularity and Complexity Profiles in Overconstrained Resource Allocation Problems," submitted to AAAI-02.

Frank, M., Bugacov, A., Chen, J., Dakin, G., Szekely, P., and Neches, R. "The Marbles Manifesto: A Definition and Comparison of Cooperative Negotiation Schemes for Distributed Resource Allocation," Proceedings of the 2001 AAAI Fall Symposium on Negotiation Methods for Autonomous Cooperative Systems, November 2-4, 2001, North Falmouth, Massachusetts.